Convegno Nazionale
Matematica senza Frontiere
Lecce, 5-8 marzo 2003

Crittografia a Chiave Pubblica

Mauro Biliotti, Gabriella Zammillo

Dipartimento di Matematica "E. De Giorgi"-Università di Lecce
biliotti@ilenic.unile.it, gabriella.zammillo@unile.it

Sommario

Il presente lavoro è dedicato alla crittografia, nello specifico, alla descrizione della tecnica del crittosistema RSA.
Teoria ed esempi, tra cui l'implementazione della tecnica con un piccolo pacchetto realizzato col software Mathematica, si alternano con l'intento di rendere più accessibile l'approccio all'argomento trattato e per mettere in evidenza l'importanza ricoperta dal ruolo delle tecnologie informatiche nel campo della ricerca matematica.

La crittografia, dal greco "cryptos": nascosto, è quell'arte o scienza che fornisce gli strumenti adatti a mantenere segrete tutte quelle informazioni che non si vogliono divulgare pubblicamente, ma rendere accessibili solo ad una o ad un ristretto numero di persone autorizzate che "sappiano come farlo".
Bisogna fare un passo indietro nel tempo di circa 2500 anni per risalire alla nascita della tecnica della crittografia, ma per recuperare uno dei più famosi crittogrammi è sufficiente arrivare al 1917 con il messaggio cifrato di Arthur Zimmerman che provocò l'ingresso degli USA nella prima guerra mondiale. Risale più o meno allo stesso periodo il cifrario ideato da Gilbert S. Vernam, sistema piuttosto sicuro, ma di difficile generalizzazione per via della chiave lunga quanto il messaggio da trasmettere, impossibile da usare una seconda volta.
L'interesse per la crittografia esplode verso la metà degli anni 70, prima con la formulazione teorica della crittografia a chiave pubblica dovuta a W. Dieffe, M.E.Hellman e R.C. Markle e poi con la sua applicazione nel 1977 dovuta a R. Rivest, A. Shamir e L. Adleman che rivoluzionano la tecnica della comunicazione segreta. A differenza di tutti gli altri sistemi crittografici infatti, gli interlocutori che utilizzano uno schema a chiave pubblica non hanno più bisogno di definire a priori una chiave segreta condivisa.
Il crittosistema, l'ambito cioè nel quale vengono effettuate le operazioni di crittazione e decrittazione, consiste di una famiglia di funzioni f (enciphering trasformations) dipendenti da parametri che all'insieme P di tutte le possibili unità di messaggio in chiaro associano l'insieme C di tutte le possibili unità di messaggio cifrato:

$$\begin{array}{ccccc} f & : & P & \longrightarrow & C \\ enciphering & & plaintext & & ciphertext \\ trasformation & & message & & message \\ & & units & & units \end{array}$$

Ma tale funzione, calcolabile conoscendo la chiave di cifratura K_E (enciphering key) che fornisce i parametri, può essere invertita conoscendo la chiave di decifratura K_D (deciphering key), consentendo così la decriptazione del messaggio.
A differenza della crittografia classica (o simmetrica) in cui K_E e K_D sono ricavabili l'una dall'altra con un algoritmo che richiede tempi di calcolo accettabili, nella crittografia a chiave pubblica, dove la coppia (f, K_E) è pubblica, ciò non vale più in quanto la funzione f^{-1} non è calcolabile in tempi accettabili se non si conosce K_D.
Siffatte funzioni f vengono chiamate *trapdoor functions* (funzioni trappola o trabocchetto). Una funzione *trappola* è in realtà, dal punto di vista matematico, una funzione biettiva e pertanto invertibile, è inoltre facile da calcolare mentre il calcolo della sua inversa è estremamente complesso.
È chiaro che l'etichetta di funzione "trabocchetto" è "provvisoria" perchè questa viene a cadere nel momento in cui si disponga di un calcolatore dalle prestazioni più elevate o magari di un algoritmo più efficiente rispetto a quelli noti.
Ad esempio nessuno ha provato finora l'esistenza di un algoritmo più efficiente di quelli noti per poter fattorizzare il prodotto di due numeri interi molto grandi, ma non si può escludere che ciò accada. Se supponiamo infatti di avere due numeri primi p e q molto grandi, intendendo con questo che siano composti da qualche centinaio di cifre, esistono algoritmi efficienti per calcolare il prodotto $n = pq$, ma non ne esistono invece per fattorizzare n e risalire a p e q.
Non è quindi un caso che per costruire funzioni trabocchetto si sia ricorsi alla fattorizzazione del prodotto di numeri interi molto grandi. Uno dei sistemi di crittografia a chiave pubblica più diffusi, noto con la denomiazione di RSA (acronimo dato dalle iniziali dei suoi inventori: Ron Rivest, Adi Shamir e Leonard Adleman) è basato infatti proprio su una semplice idea della teoria dei numeri e permette di cifrare un messaggio sfruttando il fatto che è semplice moltiplicare due numeri primi, ma è difficile fattorizzarne il prodotto.

A tal riguardo si noti che se $n = pq$, la conoscenza di n e della funzione di Eulero $\varphi(n)$, definita come il numero di interi positivi minori di n e primi con n tale cioè che:

$$\varphi(1) = 1 \quad e \quad \varphi(n) = |\{a \in N : 0 < a \leq n,\, (a, n) = 1\}|,$$

è equivalente alla conoscenza dei numeri p e q avendosi:

$$n = pq \quad e \quad \varphi(n) = (p-1)(q-1) = n - (p+q) + 1$$

da cui:

$$pq = n \quad e \quad (p+q) = n - \varphi + 1$$

e quindi p e q risultano essere soluzioni dell'equazione di 2^0 grado:

$$x^2 + (\varphi(n) - n - 1)x + n = 0.$$

Se quindi l'utente A volesse ricevere un messaggio crittografato da B ricorrendo alla tecnica dell'RSA, A dovrà generare le sue chiavi e pertanto sceglierà prima di tutto due numeri primi p e q molto grandi tali che il loro prodotto sia pari ad n. Successivamente sceglierà un numero $d < \varphi(n)$ che sia primo con $\varphi(n)$ e calcolerà il numero e tale che:

$$ed = 1 mod[\varphi(n)]$$

La chiave pubblica di A sarà data dalla coppia (e, n) mentre quella privata dalla coppia (d, n), ma solo d è segreta. B, che conosce la chiave pubblica di A, crittograferà il messaggio e lo invierà ad A che a sua volta potrà decifrarlo perché in possesso della propria chiave segreta.

La codifica del messaggio espresso attraverso il numero $M < n$ ed inviato dall'utente B all'utente A sarà dato da:

$$E(M) = M^e mod\, n$$

ossia $E(M)$ definisce la *encryption function.*
L'utente A e soltanto A, con la sua chiave segreta, può decriptare il messaggio poiché risulta:

$$M = D[E(M)] = [E(M)]^d mod\, n$$

È evidente che l'operazione di cifratura E e decifratura D sono una l'inversa dell'altra ed è proprio il teorema di Eulero a garantire che se $M < n$ allora $D[E(M)] = M$.
Si osservi che d può essere ricavato solo conoscendo $\varphi(n)$ ossia, per quanto visto, solo riuscendo, dato n, a determinare i suoi fattori p e q.

Giusto perché ci si possa rendere conto di come funzioni la tecnica dell'RSA, vediamo un esempio utilizzando numeri piuttosto piccoli.
Supponiamo che l'utente A scelga $p = 47$ e $q = 71$.
Calcola quindi:

$$n = pq \quad \text{e} \quad \varphi(n) = (p-1)(q-1)$$

ottenendo così:

$$n = 3337 \quad \text{e} \quad \varphi(n) = 3220$$

Sceglie il numero e tale che il $m.c.d.\,(e, \varphi(n)) = 1$.
Supponiamo sia $e = 79$ e successivamente calcola:

$$d = e^{-1} mod\, \varphi(n)$$

ottenendo:

$$d = 79^{-1} mod\, 3220 = 1019.$$

Poiché la chiave pubblica è data dalla coppia (e, n), sarà:

$$(e, n) = (79,\ 3337)$$

mentre la chiave privata sarà la coppia

$$(d, n) = (1019,\ 3337).$$

L'utente B, a conoscenza della chiave pubblica di A, supponiamo voglia inviare ad A il messaggio $M = 688$, dove M rappresenta una sequenza di numeri ottenuta dalla conversione delle lettere del testo del messaggio nei corrispondenti valori del codice ascii.
B cripta quindi il testo calcolando:

$$E(M) = M^e mod\, n = 688^{79} mod\, 3337 = 1570$$

che l'utente A, dopo averlo ricevuto, decripta ricorrendo alla sua chiave segreta e applicando la formula:

$$M = D[E(M)] = [E(M)]^d mod\, n$$

vale a dire: $M = 1570^{1079} mod\, 3337 = 688$.

Per un calcolo realistico in cui il numero n da fattorizzare sia composto da parecchie cifre, è necessario affidarsi ad un potente software.
Nell'esempio riportato di seguito (R.Caballero, J.Rojo -1996) possiamo osservare come con un semplice pacchetto realizzato col programma Mathematica, sia possibile implementare la tecnica dell'RSA passando dalla generazione dei due numeri primi e quindi dalla generazione delle chiavi pubbliche e private degli interlocutori A e B, alla cifratura del messaggio in stringa di numeri attraverso il codice ascii; dalla suddivisione della stringa in sottostringhe, nel caso in cui il messaggio dovesse risultare piuttosto lungo, alla decodifica di quest'ultimo una volta giunto a destinazione.

Cominciamo quindi scegliendo un numero a e verifichiamo che sia primo.

```
Clear

NextPrime[a_Integer]:=a  /;PrimeQ[a];
NextPrime[a_Integer]:=NextPrime[a+2]/;OddQ[a];
NextPrime[a_Integer]:=NextPrime[a+1]/;EvenQ[a];
```

Trovato il primo sufficientemente grande chiamato pa, ne scegliamo un altro che chiamiamo qa e poiché anche questo deve risultare primo, si sfrutta la funzione precedentemente data.

```
pa=NextPrime[832946567513390228562041863202631208065435134 6
    501674980653132165406540321984046546540654984654326543]

qa=NextPrime[365498064562197806546213468460321648905195187 4
    374910951262379516512162162198484576370259654654210319]
```

Calcoliamo il prodotto dei due primi:

```
na=pa qa
```

e troviamo un numero d che sia primo con il prodotto $\varphi(n) = (p-1)(q-1)$.

```
NextNumber[a_Integer,p_Integer,q_Integer]:=
    a /;GCD[a,(p-1)(q-1)]\[Equal]1;
NextNumber[a_Integer,p_Integer,q_Integer]:=
NextNumber[a+1,p,q];
da=NextNumber
[59832198498732165498132065406498406213203210656879089048 7
        078108470928734182718493984754934045886969 7,pa,qa]
```

Calcoliamo il numero e come inverso di d tale che $ed = 1 \mod \varphi(n)$

```
RSA[d_Integer,p_Integer,q_Integer]:=
    PowerMod[d,-1,(p-1) (q-1)];

ea=RSA[da,pa,qa]
```

Per poter applicare la procedura di autenticazione del messaggio, descritta nel seguito, è necessario che anche B esegua le stesse operazioni calcolando pb, qb, nb, db, eb e rendendo pubblico eb ed nb che rappresentano le sue chiavi pubbliche. Per B quindi otteniamo:

```
pb=NextPrime[492748374593785673829191893784375683939588757 3
   839398477584939393575369452973951752841544156566516359];

qb=NextPrime[124903852093485203956802389467023948570234562 3
   749546734623590623497342891891129387724378439580249837];

nb=pb qb;

db=NextNumber[489393578569697785763436859696473648950658 3
    773219643400765914612004074213453983215672215739397576
    943,pb,qb];

eb=RSA[db,pb,qb];
```

Trasformiamo ora il testo del messaggio in una stringa di numeri attraverso il codice ascii.

```
AsciiToCode[x_Integer]:=
    If[x<132 && x>31, x-31, 1];

StringToNumber[a_String]:=
  Fold[100 #1+#2 &,0,Map[AsciiToCode,
      ToCharacterCode[a]]]
```

A titolo di esempio trasformiamo in stringa di numeri l'espressione "Matematica senza frontiere":

```
number=StringToNumber["MATEMATICA SENZA FRONTIERE"]
```

Procedendo al contrario, riusciamo a trasformare la stringa di numeri in una stringa di caratteri attraverso la funzione:

```
CodeToAscii[x_Integer]:=
    If[x<100 && x>0, x+31, 32];
NumberToString[a_Integer]:=
    FromCharacterCode[Reverse[Map[CodeToAscii[#1-100
                  Floor[#1/100]] &,
      Drop[FixedPointList[(Floor[#1/100] &),a],-2]]]];

NumberToString[number]
```

Nel caso in cui il messaggio da codificare e decodificare risultasse troppo lungo, si può pensare di suddividere la stringa in sottostringhe di lunghezza minore di un certo n, con n fissato arbitrariamente.

```
Codify[a_String,e_Integer,n_Integer]:=
    PowerMod[StringToNumber[a],e,n]/;StringToNumber[a]<n;
Codify[a_String,e_Integer,n_Integer]:=
    Print["Message too long"];

StringTakeRSA[a_String,i1_Integer,i2_Integer]:=
    StringTake[a,{i1,i2}]/;i2<=StringLength[a];
StringTakeRSA[a_String,i1_Integer,i2_Integer]:=
    StringTake[a,{i1,StringLength[a]}]/;
    i2>StringLength[a];

CodifyText[a_String,e_Integer,n_Integer]:=
  Module[{b,i},
    b=Table[StringTakeRSA[a, i, i+Floor[N[Log[10,n]]/2]-1],
    {i,1,StringLength[a],Floor[N[Log[10,n]]/2] } ];
    Return[Table[Codify[b[[i]],e,n],{i,1,Length[b]}] ];
    ]
```

Tale funzione genera una lista di numeri ognuno corrispondente ad una parte del messaggio. È un'operazione eseguibile da chiunque perché richiede

soltanto l'utilizzo della chiave pubblica. Nel caso in cui B volesse spedire un messaggio ad A, non dovrà che digitare comando e testo.

```
list=CodifyText[
    "Cara A, dal 5 all'8 marzo prossimo si terra' a Lecce
    il Convegno Nazionale MATEMATICA SENZA FRONTIERE.
    Non mancare B.",ea,na]
```

In particolare, l'utente B, nel caso in cui A voglia avere la certezza che il messaggio sia stato spedito effettivamente da B, ricorrerà alla tecnica della segnatura. È un'operazione di autenticazione basata sul seguente metodo: sia M il messaggio da spedire; Ea e Da la chiave pubblica e privata di A; Eb e Db la rispettiva chiave pubblica e privata di B.
B spedisce il messaggio $Ea(Db(M)) = N$ applicando prima la propria chiave privata e dopo quella pubblica di A.
L'utente A, a sua volta, legge $M = Eb(Da(N))$ applicando prima la sua chiave privata e poi quella pubblica di B.
Ovviamente, solo B può aver spedito N perchè solo lui conosce la propria chiave privata.

```
CodifyTextWithSignature[a_String,d1_Integer,
  n1_Integer,e2_Integer,n2_Integer]:=
  PowerMod[CodifyText[a,d1,n1],e2,n2];
```

Per la decodifica si procede esattamente al contrario.

```
Decodify[a_Integer,d_Integer,n_Integer]:=
    NumberToString[PowerMod[a,d,n]];

DecodifyList[a_List,d_Integer,n_Integer]:=
    StringJoin[Table[Decodify[a[[i]],d,n],{i,1,Length[a]}]];

DecodifyListWithSignature[a_List,e1_Integer,n1_Integer,
   d2_Integer,n2_Integer]:=
   DecodifyList[PowerMod[a,d2,n2],e1,n1];

DecodifyList[list,da,na]
```

A titolo di esempio vediamo un'applicazione della tecnica della segnatura. Supponiamo che B voglia spedire un messaggio ad A, che lo stesso A può leggere ed è sicuro che provenga da B.
B non conosce la chiave segreta Da di A ed A non conosce la chiave segreta Db di B. B scrive:

```
listsigned=CodifyTextWithSignature[
    "Cara A, dal 5 all'8 marzo prossimo si terra' a Lecce il
    Convegno Nazionale MATEMATICA SENZA FRONTIERE.
    Non mancare B.",db,nb,ea,na];
```

L'utente A per leggere il messaggio deve semplicemente decodificare col seguente comando:

```
DecodifyListWithSignature[listsigned,eb,nb,da,na]
```

Se l'RSA permette di cifrare un messaggio sfruttando il fatto che è semplice moltiplicare due numeri primi molto grandi, ma alquanto complicato fattorizzarne il prodotto, non è certamente fuori luogo chiedersi come verificare che il numero scelto sia effettivamente primo.
I metodi usati sono per la maggior parte probabilistici.
I tests di primalità sono infatti dei metodi che ci permettono di stabilire se un dato numero n sia primo o meno con probabilità prefissata. In pratica, un numero n è considerato primo se supera un determinato numero di tests. Per il Piccolo Teorema di Fermat è noto che se n è un primo e b un intero che sia primo con n, allora:

$$b^{n-1} \equiv 1 \, mod \, n \tag{1}$$

Nel caso in cui n non risultasse essere primo, ma un intero positivo dispari e composito e b un intero tale che $0 < b < n$, primo con n, per cui dovesse continuare a valere la (1), allora n si dirà uno *pseudoprimo* rispetto alla base b.

È dimostrabile che se il test fallisce per una base, allora fallisce per almeno la metà delle basi e, di conseguenza, k tests superati assicurano che la probabilità che n sia composito non è superiore a $1/2^k$. Purtroppo però esistono dei numeri compositi che superano il test per ogni base (numeri di Carmichael).

Un test più sicuro è il seguente: sappiamo che se n è un primo, b un intero primo con n e (b/n) il simbolo di Jacobi, allora vale:

$$b^{(n-1)/2} \equiv (b/n) \, mod \, n. \tag{2}$$

Se n è un numero dispari composito, b un intero primo con n e vale la (2) o, come si dice, se n *è uno pseudoprimo di Eulero rispetto alla base* b, allora la relazione (2) è falsa per almeno il 50% di tutti i b primi con n, ossia il test fallisce per almeno la metà delle basi. Quindi k tests superati danno una probabilità che n sia composito non superiore a $1/2^k$.

La funzione Next Prime, richiamata col pacchetto di Mathematica nell'esempio di implementazione del criptosistema, consente di determinare in tempi rapidi se i numeri p e q, scelti per il nostro scopo, siano primi o meno.

Riferimenti bibliografici

[1] N. Koblitz. A course in number theory and cryptography *Springer*, 1994

[2] C.H. Bennett, G. Brassard, A.K. Ekert. Crittografia quantistica *Le Scienze*, n.292, pp. 88-96, 1992

[3] R. Caballero, J. Rojo. An efficient implementation of the RSA Cryptographic Technique. (in Actas de la IV Reunion Española sobre Criptologia) *J. Tena Y M.F. Blanco*, pp. 109-116, 1996

[4] http://www-math.science.unitn.it/∼caranti/Didattica/Algebra2...

[5] http://www.dia.unisa.it/ads.dir/corso-security/www/Corso-9900/rsa...